CLAIMS

Having thus described our invention, what we claim as new, and desire to secure by Letters Patent is:

1. A method for mapping a valid stack up to a destination program counter, comprising:

mapping a path of control flow on the stack from any start point in a selected method to the destination program counter; and

simulating stack actions for executing bytecodes along said path.

2. The method of claim 1 wherein the step of mapping a path of control flow on the stack comprises:

processing a first linear bytecode sequence until the control flow is interrupted; and recording unprocessed targets from any branches in the first linear bytecode sequence for future processing.

3. The method of claim 2 wherein the step of mapping a path of control flow on the stack further comprises:

processing an additional bytecode linear sequence until the control flow is interrupted; and

recording unprocessed targets from any branches in the additional linear bytecode sequence for future processing, where the destination program counter was not reached during an earlier processing of a linear bytecode sequence.

4. The method of claim 2 wherein the step of processing any linear bytecode sequence comprises:

determining if a bytecode in said any linear bytecode sequence is a breakpoint with a pointer to bytecode data; and

replacing the breakpoint with the bytecode data.

5. The method of claim 3 wherein the step of processing any linear bytecode sequence comprises:

determining if a bytecode in said any linear bytecode sequence is a breakpoint with a pointer to bytecode data; and

replacing the breakpoint with the bytecode data.

Sub C3

6. The method of claim 1 wherein the step of simulating stack actions executing the bytecodes along the path further comprises generating a virtual stack.

7. The method of claim 6, further comprising:

encoding the virtual stack as a bitstring and storing the bitstring at a selected destination for use in memory management operations.

8. The method of claim 7, wherein the step of storing the bitstring comprises storing the bitstring to the selected method as compiled on a heap.

9. The method of claim 7, wherein the step of storing the bitstring comprises storing the bitstring to a pre-allocated area on the stack.

10. The method of claim 1 wherein the step of simulating stack actions executing the bytecodes along the path further comprises:

inserting pre-determined stack actions for bytecodes maintaining the control flow in the selected method; and

calculating stack actions for bytecodes transferring the control flow from the selected method.

11. A method for mapping a Java bytecode stack up to a destination program counter comprising:

mapping a path of control flow on the stack from any start point in a selected method to the destination program counter; and

simulating stack actions for executing bytecodes along said path.

12. The method of claim 11 wherein the step of mapping a path of control flow on the stack comprises:

processing a first linear bytecode sequence until the control flow is interrupted; and recording unprocessed targets from any branches in the first linear bytecode sequence for future processing.

13. The method of claim 12 wherein the step of mapping a path of control flow on the stack further comprises:

processing an additional bytecode linear sequence until the control flow is interrupted; and

recording unprocessed targets from any branches in the additional linear bytecode sequence for future processing, where the destination program counter was not reached during an earlier processing of a linear bytecode sequence.

14. The method of claim 12 wherein the step of processing any linear bytecode sequence comprises:

determining if a bytecode in said any linear bytecode sequence is a breakpoint with a pointer to bytecode data; and

replacing the breakpoint with the bytecode data.

15. The method of claim 13 wherein the step of processing any linear bytecode sequence comprises:

determining if a bytecode in said any linear bytecode sequence is a breakpoint with a pointer to bytecode data; and

replacing the breakpoint with the bytecode data.

Sub C3

16. The method of claim 11 wherein the step of simulating stack actions executing the bytecodes along the path further comprises generating a virtual stack.

17. The method of claim 16 further comprising:

encoding the virtual stack as a bitstring and storing the bitstring at a selected destination for use in memory management operations.

18. The method of claim 17, wherein the step of storing the bitstring comprises storing the bitstring to the selected method as compiled on a heap.

19. The method of claim 17, wherein the step of storing the bitstring comprises storing the bitstring to a pre-allocated area on the stack.

20. The method of claim 11 wherein the step of simulating stack actions executing the bytecodes along the path further comprises:

inserting pre-determined stack actions for bytecodes maintaining the control flow in the selected method; and

calculating stack actions for bytecodes transferring the control flow from the selected method.

21. A computer-readable memory for storing the instructions for use in the execution in a computer of the method of claim 1.

22. A computer readable memory for storing the instructions for use in the execution in a computer of the method of claim 11.

23. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for mapping a valid stack up to a destination program counter, said method steps comprising:

mapping a path of control flow on the stack from any start point in a selected method to the destination program counter; and

simulating stack actions for executing bytecodes along said path,

wherein the step of mapping a path of control flow on the stack comprises:

processing a first linear bytecode sequence until the control flow is interrupted; and

recording unprocessed targets from any branches in the first linear bytecode sequence for future processing, and

where the destination program counter was not reached during an earlier processing of a linear bytecode sequence,

processing an additional bytecode linear sequence until the control flow is interrupted; and

recording unprocessed targets from any branches in the additional linear bytecode sequence for future processing.